15/09/2020

JAVASCRIPT CONCEPTS CONTINUATION

JavaScript continue Statement:

- The continue statement breaks one iteration in the loop if a specified condition occurs, and continues with the next iteration in the loop.
- The difference between continue and the break statement, is instead of "jumping out" of a loop, the continue statement jumps over one iteration in the loop.

  For example:

```
<!DOCTYPE html>
<html>
<body>
<h2>Javascript continue statement</h2>
<script>
  var cars = ["Audi", "bmv", "ferrari", "Ford"];
var text = "";
  var i;
  for (i = 0; i < cars.length; i++) {
    if (cars[i] === "bmv") {
      continue;
```

```
    }
    text += cars[i] + "<br>";
  }
  document.write(text);
</script>
</body>
</html>
```

String methods continuation:

1. Converting to Upper and Lower Case

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert string to upper case</h2>
<script>                                    //Strings are immutable that means Strings cannot  be changed, only replaced.
 var text = "sandeep";
 document.write(text.toUpperCase());//  For lowercase use toLowerCase();
</script>
</body>
</html>
```

2. String.trim()

- The trim() method removes whitespace from both sides of a string

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript trim</h2>

<script>

var str = "     Sandeep     ";
document.write((str.trim()));

</script>
</body>
</html>
```

3. The charAt() Method:

- The charAt() method returns the character at a specified index (position) in a string:

```
<!DOCTYPE html>
<html>
<body>
<h2>charAt() method</h2>
<script>
var str = "Sandeep";
document.write(str.charAt(2));
</script>
```

```
</body>
</html>
```

4. Converting a String to an Array

- A string can be converted to an array with the split() method:

```
<!DOCTYPE html>
<html>
<body>
<script>
var str = "SWENG";
var arr = str.split("");
var text = "";
var i;
for (i = 0; i < arr.length; i++) {
  text += arr[i] + "<br>"
}
document.write(text);
</script>
</body>
</html>
```

JAVASCRIPT COLLECTIONS

- Any group of individual objects which are represented as a single unit is known as the collection of the objects.
- In the high-level languages like Java and JavaScript, we don't need to explicitly allocate or release memory.
- JavaScript values are allocated when things are created (objects, Strings, etc.) and freed automatically when they are no longer used. This process is called Garbage collection.
- JavaScript Garbage Collection is a form of memory management whereby objects that are no longer referenced are automatically deleted and their resources are reclaimed. Map and Sets references to objects are strongly held and will not allow for garbage collection.

JAVASCRIPT ARRAY METHODS

1. Converting Arrays to Strings

- The JavaScript method toString() converts an array to a string of comma separated array values.

```
<!DOCTYPE html>
```

```
<html>
<body>

<h2>JavaScript Array Methods</h2>

<h2>toString()</h2>

<script>
var fruits = ["pineapple", "Orange", "Apple", "Mango"];
document.write(fruits.toString());
</script>
</body>
</html>
```

2. Popping and Pushing:

- The pop() method removes the last element from an array

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Array Methods</h2>

<h2>pop()</h2>

<script>
var fruits = ["pineapple", "Orange", "Apple", "Mango"];
document.write(fruits.pop());
document.write("<br>");
document.write(fruits);
```

```
</script>
</body>
</html>
```

- The push() method adds a new element to an array at the end.
- The push() method returns the new array length.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Array Methods</h2>

<h2>push()</h2>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("watermelon"));
document.write("<br>");
document.write(fruits);
</script>
</body>
</html>
```

3. Shifting Elements

- Shifting is equivalent to popping, working on the first element instead of the last.

```
<!DOCTYPE html>
<html>
```

```
<body>

<h2>JavaScript Array Methods</h2>

<h3>shift()</h3>

 <script>

var fruits = ["pineapple", "Orange", "Apple", "Mango"]
document.write(fruits.shift());
document.write("<br>");
document.write(fruits);
</script>
</body>
</html>
```

4. Splicing an Array

- The splice() method can be used to add new items to an array: For example fruits.splice(2, 0, "Lemon", "Kiwi");
- The first parameter (2) defines the position where new elements should be added (spliced in).
- The second parameter (0) defines how many elements should be removed.
- The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Array Methods</h2>
<h2>splice()</h2>
```

```
<script>
var fruits = ["Pineapple", "Orange", "Apple", "Mango"];
document.write("Original Array:<br>" + fruits);
  fruits.splice(2, 0, "Lemon", "Kiwi");
  document.write("<br>");
  document.write("New Array:<br>" + fruits);

 </script>

</body>
</html>
```

Javascript constructors:

- The constructor() method is a special method for creating and initializing objects created within a class.
- the default constructor is created by using constructor().
- The constructor() method is called automatically when a class is initiated, and it has to have the exact name "constructor", in fact,
- if we do not have a constructor method, JavaScript will add an invisible and empty constructor method.

  For example,

  ```
  <!DOCTYPE html>
  <html>
  <body>
  <script>
  ```

```
class employee{
constructor(){
            this.id=22;
   this.name = "Sandeep";
  }
}
var emp = new employee();
document.write(emp.id+" "+emp.name);
</script>
</body>
</html>
```

Parameterized constructor and super method

- The super keyword is used to call the parent class constructor.

```
<!DOCTYPE html>
<html>
<body>

<script>
class CompanyName
{
  constructor()
```

```
  {
    this.language="Javascript";
  }
}

class Employee extends CompanyName {
  constructor(id,name) {
   super();
    this.id=id;
    this.name=name;
  }
}
var emp = new Employee(1,"sandeep");
document.write(emp.id+" "+emp.name+" "+emp.language);
</script>
</body>
</html>
```